

Dialectic: Enhancing Text Input Fields with Automatic Feedback to Improve Social Content Writing Quality

Hamed Nilforoshan, James Sands, Kevin Lin, Rahul Khanna, Eugene Wu

Columbia University

{hn2284, js4597, kl2806, rk2658}@columbia.edu, ewu@cs.columbia.edu

Abstract

Modern social media relies on high quality user generated writing such as reviews, explanations, and answers. In contrast to standard validation to provide feedback for structured inputs (e.g., dates, email addresses), it is difficult to provide timely, high quality, customized feedback for free-form text input. While existing solutions based on crowdsourced feedback (e.g., upvotes and comments) can eventually produce high quality feedback, they suffer from high latency and costs, whereas fully automated approaches are limited to syntactic feedback that does not address text content. We introduce Dialectic, an end-to-end extensible system that simplifies the process of *creating, customizing, and deploying* content-specific feedback for free-text inputs. Our main observation is that many services already have a corpus of crowdsourced feedback that can be used to bootstrap a feedback system. Dialectic initializes with a corpus of annotated free-form text, automatically segments input text, and helps developers rapidly add domain-specific document quality features as well as content-specific feedback generation functions to provide targeted feedback to user inputs. Our user study shows that Dialectic can be used to create a feedback interface that produces an average of 14.4% quality improvement of product review text, over 3x better than a state-of-the-art feedback system.

Introduction

Modern social media relies on user generated text: product websites (e.g., Amazon) rely on user submitted product reviews to help users make purchasing decisions; Q&A services (e.g., StackOverflow, reddit, Quora) rely on the availability of high quality user generated questions and answers. Their success is directly related to the quality of the content shown to users, and a key challenge is to manage and improve the quality of the user generated text content that they serve.

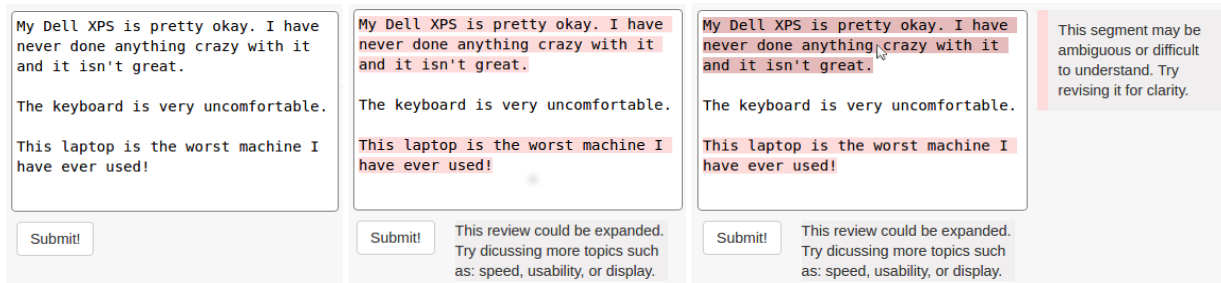
The primary quality control methods are filtering to remove/hide low quality or spam content (Spirin and Han 2012) and ranking to prioritize the user-generated content to serve by using user-based ratings (Tang, Hu, and Liu 2013; Guy 2015) or automatic rankings (Agichtein et al. 2008; Wang et al. 2013; Yang and Amatriain 2016; Siersdorfer et al. 2010). These approaches fundamentally assume that the corpus of user contributions is sufficiently large; however, the long tail of entities (e.g., less popular products, esoteric

questions) will not always have enough user generated content to contain high quality documents to serve. In these cases, it is more important to improve the quality of *every* user contribution.

To this end, incentives such as badges and enhanced user status (Wang et al. 2013; Bosu et al. 2013) encourage users to provide higher quality contributions, and seek to attract and retain high quality contributors. This can even include material goods: Yelp Elite offers top contributors access to exclusive events with free food and drink (Yelp 2016a). However, incentives are general to the overall service rather than to any specific user input (e.g., a product review or answer), and may still fail to address the quality of long tail content.

Rapid feedback during the writing process improves content quality (Kulik and Kulik 1988). Recent systems use crowdsourcing that trains peers to generate customized feedback for student writing assignments (Kulkarni, Bernstein, and Klemmer 2015); however, the feedback latency is nearly 20 minutes, too long for many online contexts. Fully automated systems can be instantaneous, but are limited to grammar, misspellings and other syntactic errors (Madnani and Cahill 2014; Microsoft 2016; Google 2016).

Learning-based methods are a promising middle-ground (Krause 2015b; Biran and McKeown 2014; Krause, Perer, and Ng 2016) that identify feature values that are indicative of low quality text and translate them into feedback to help improve the content. However, two factors limit the feedback quality of existing approaches; feedback addresses features in isolation without taking interactions between different features into account, and is often limited to broad suggestions at the full document granularity (Krause 2015b; Boomerang 2016; FoxType 2016). In contrast, text-quality prediction work highlights the value of accounting for multi-feature interactions (Weimer, Gurevych, and Mühlhäuser 2007; Ghose and Ipeirotis 2011), and feedback psychology studies show that localized, targeted feedback is crucial (Nelson and Schunn 2007), but these benefits have yet to be integrated into a feedback generation system. Finally, despite research in new predictive features and feedback generation, it is simply difficult to build, customize, and deploy an end-to-end feedback systems to test new ideas, and to sustain such systems in light of new findings in these areas.



(a) Non-annotated review text. (b) Document feedback is shown below the text. (c) Hovering over highlights shows segment specific feedback.

Figure 1: Example of Dialectic feedback interface.

To this end, Dialectic is a system to easily build and deploy new writing feedback systems. It automates common tasks such as text segmentation, training, and deployment, so developers can focus on domain-specific tasks such as creating features and generating feedback explanation text. Although many components are well known, their combination into a user-facing extensible system (Figure 1) does not exist. In addition, our user study shows that combining segmentation and our novel perturbation-based feedback generates more effective writing feedback, compared to applying either technique separately to alternative state-of-the-art systems.

Developers first provide a corpus of labeled user generated text (e.g., helpfulness votes for product reviews, or up/down votes for comments). Offline, Dialectic automatically segments the text documents based on a configurable segmentation method (by default, we segment by topic), and trains document and segment-level quality prediction models. To generate feedback for a text input, Dialectic uses the models to find low-quality text, and uses a novel perturbation-based technique to identify *combinations of features* that affect the writing quality. Developer-provided explanation functions map these features into feedback text. As new features are identified in the literature, they can easily be incorporated. In our evaluation, Dialectic generated feedback for multiparagraph text inputs (reviews ranging from 1-5 paragraphs) in < 1 second.

In summary, our main contributions are:

- The design and implementation of Dialectic, an extensible text feedback system that combines text segmentation, classification, and explanation to automate both broad document-level and granular segment-level feedback on user generated writing. To use the system, it can be installed using the python package manager `pip anonymized for submission`.
- A novel perturbation-based technique that identifies combinations of features that, if changed, will most improve the predicted quality of input text. We formalize this problem, and propose an efficient heuristic to search the exponential solution space that empirically produces helpful feedback.
- End-to-end evaluation by training Dialectic on Amazon product reviews using existing features from the literature, and conducting a crowd-sourced user study to ex-

amine how automated feedback improves product review writing. Combining our perturbation technique to generate *segment-specific* feedback improves average review quality by 14.4%, and over 3x more than a baseline based on state-of-the-art feedback generation.

Related Work

Our research focuses on a system for providing targeted writing feedback to improve user text contributions on websites that rely on user-generated text content. The success of such websites partially depends on presenting high quality content and user contributions (Archak, Ghose, and Ipeirotis 2011; Li, Ghose, and Ipeirotis 2011; Ghose, Ipeirotis, and Sundararajan 2007; Ghose and Ipeirotis 2009). As such, the related work falls into several categories: post-hoc methods that maximize the quality of surfaced content through filtering and ranking, indirect mechanisms to improve user generated writing quality, and direct feedback-oriented mechanisms to improve writing quality.

Post-hoc Approaches: Post-hoc approaches filter poor content (Spirin and Han 2012) such as spam; sort and surface higher quality content (Tang, Hu, and Liu 2013; Guy 2015; Agichtein et al. 2008) such as product reviews (Mudambi and Schuff 2010), answers to user comments (Wang et al. 2013; Yang and Amatriain 2016), or forum comments (Siersdorfer et al. 2010); or edit user reviews for clarification or grammatical purposes (Ipeirotis 2016). These approaches assume a large corpus that contains high quality content for every topic (e.g., product or question). In reality, there is often a long tail of topics without sufficient content for such approaches to be effective (Saito 2016; McAuley, Pandey, and Leskovec 2015). For such cases, improving quality upstream during user input process may be more effective.

Indirect Mechanisms: Indirect methods such as community standards and guidelines (Nov 2007; Bakshy, Karrer, and Adamic 2009; Amazon 2016) help clarify quality standards, while up-votes and ratings provide social incentives (Muchnik, Aral, and Taylor 2013; Bosu et al. 2013). Incentive mechanisms such as badges, scores (Ghosh 2012; Deterding et al. 2011), status (Zappos 2016), or even money (Kim 2012; Ipeirotis 2016) have also been used to keep good contributors. These methods focus more on finding good con-

tributors and lack content-specific feedback (e.g., discuss camera quality for a phone).

Direct Writing Feedback: We focus on feedback interfaces to improve text quality during the writing process. Crowd-based feedback has been shown to improve writing quality but can take 20 minutes to generate feedback. (Kulkarni, Bernstein, and Klemmer 2015)—existing research emphasizes the importance and benefits of immediate writing feedback (Kulik and Kulik 1988) and suggests the value of automated approaches. Although deep semantic feedback is automated in code development environments (Roy Choudhury, Yin, and Fox 2016; Singh, Gulwani, and Solar-Lezama 2012; Rivers and Koedinger 2014), the same techniques cannot be generalized from highly constrained code grammars to free-form human text.

Dialectic uses a two-step automated approach to generate feedback for the entire document’s text as well as individual document segments. It first uses models to identify low quality text and then generates targeted feedback for that text. The first step is motivated by automatic essay graders, which use predictive models to assign overall quality scores to written content (Valenti, Neri, and Cucchiarelli 2003; Farra, Somasundaran, and Burstein 2015; Attali and Burstein 2004; Madnani and Cahill 2014). Numerous predictive features have been studied across text domains such as code reviews (Krause 2015a), YouTube comments (Siersdorfer et al. 2010), Amazon product reviews (Ghose and Ipeirotis 2011; Kim et al. 2006), movie reviews (Liu et al. 2008), and reddit comments (Tan et al. 2016). Rather than innovate on features, we survey and include a broad range of default features from prior work in Dialectic, which we describe in the **Predicting Low-quality Text** subsection of the Implementation section.

The second step is generating feedback that suggests changes to *most improve* the predicted text quality. A common method is to identify individual features that are outliers from “typical” feature values in high quality documents, and map these outlier features to pre-written feedback text (Krause, Perer, and Ng 2016; Boomerang 2016; Biran and McKeown 2014). However, existing techniques use linear models that both ignore multi-feature interactions, and are often less predictive of quality than Random Forests (Weimer, Gurevych, and Mühlhäuser 2007; Ghose and Ipeirotis 2011), which are able to account for multi-feature interactions. Consider a product review containing a long, angry diatribe about customer service. By studying sentiment and length features in isolation, existing systems may suggest reducing length and decreasing emotion. However, such systems would not recognize that the review can be most improved by simultaneously reducing the emotion in the text and *increasing* the length by including more details about the product.

There is related work that seeks to explain a model’s predictions by e.g., generating a sparse more understandable model (Ribeiro, Singh, and Guestrin 2016) or identifying subsets of the input sufficient to maintain a model’s prediction (Lei, Barzilay, and Jaakkola 2016). These explanation approaches are complementary to our goals of sug-

gesting *improvements* to the text quality. Our work focuses on white-box random forest models and explores novel ways to *change* the model’s prediction by perturbing the input feature vector. To the best of our knowledge, we present the first attempt to generate suggestions on how to improve the utility of a specific document by examining the global space of possible alternative classifications in a random forest model.

System Design

A Segmentation Oriented Approach: Dialectic is an automated feedback generation system that augments traditional document-level feedback with localized segment-level feedback. Existing research on writing feedback shows that localized feedback is more helpful because it increases the likelihood that feedback recipients will identify and improve the specific issue (Nelson and Schunn 2007). In online writing settings, feedback in the form of comments are more effective at motivating revision than a standardized rubric (Kulkarni, Bernstein, and Klemmer 2015). Thus, in order to provide targeted and free-form feedback to users, Dialectic first segments the input document and then suggests changes on a segment-level, in addition to providing general document-level feedback. Figure 2 summarizes the overall data flow to generate segment-level feedback. Note that the classification and feedback occurs for both low quality segments as well as for the entire document; however, our discussion will focus on the benefits and implementation of segment-level feedback.

Architecture Overview: Figure 3 depicts the Dialectic system architecture, which consists of offline and online components. The offline components (blue arrows) take as input a corpus of training data in the form of user generated text documents and their labels—for instance, Amazon product reviews may be labelled by the ratio of “helpful” and “unhelpful” votes. The *Segmenter* first splits each document into segments. The *Model Generator* then trains a collection of classification models to predict the quality of a user’s overall text submission as well as its constituent segments; these are cached in the *Model Store*. The online components (green arrows) send the contents of a text input widget, along with an optional corpus name, to the web-server. Dialectic uses the models in the *Model Store* to identify whether the entire document and/or segments generated by the *Segmenter* are low quality. The *Feedback Generator* then constructs feedback explanations for the low quality text, which are returned and displayed in the text widget.

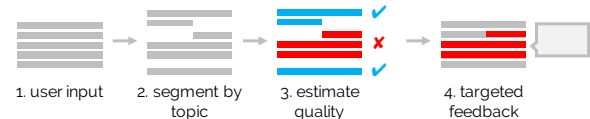


Figure 2: Dialectic splits user input into coherent segments; estimates the quality of each segment and the text as a whole; and generates and shows suggested improvements to the user.

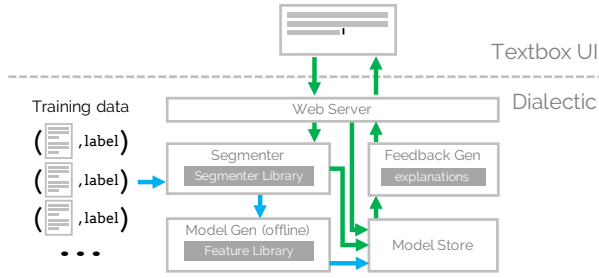


Figure 3: Dialectic online and offline architecture. **Blue** arrows depict the offline training and storage process. **Green** arrows depict the online execution flow when a user submits.

Implementation

The input to Dialectic is a labeled corpus of user-generated text documents (e.g., product reviews and their helpfulness ratings), which the system uses to train document and segment-level prediction models. The models are used to generate feedback for new user input text. To generate feedback, we assume a utility function $U : \mathbb{N} \rightarrow \mathbb{R}$ over the classification labels and seek to identify changes to the user input that will maximize the expected utility function output. For instance, our evaluations use binary helpful/unhelpful labels with utilities 0 and 1, respectively. We also assume that the user has provided \mathcal{E} , a set of domain-specific *explanation functions* $e : \mathcal{F} \times \mathbb{R}^n \rightarrow \text{text}$ which take as input a set of features and their values for a given text input, and return a text-feedback (see **Feature Generator** subsection in Explanation Functions). In short, Dialectic identifies features of the input document (or its segments) that, if changed, will most improve its utility, and uses the explanation functions to transform these features into feedback text.

Segmenter

The segmenter supports any segmentation algorithm that splits multi-paragraph text into an array of text segments. Contributor rubrics across many social media services are structured around topics (Yelp 2016b; Amazon 2016; Wikipedia 2016), and psychology research suggests that mentally processing the topical hierarchy of text is fundamental to the reading process (Hyönä, Lorch Jr, and Kaakinen 2002). Thus, by default we segment and critique documents at topic-level units.

To this end, we use a technique called TopicTiling (Riedl and Biemann 2012), an extension to TextTiling (Hearst 1997), that uses a sliding window approach to compute the LDA topic distribution within each window and create a new segment when the distribution changes beyond a threshold. TopicTiling outperformed other topic segmenters (Misra et al. 2011; Ji-Wei Wu 2011) in terms of their WindowDiff score (Pevzner and Hearst 2002) on our hand-segmented, test corpi.

Developers can easily add custom segmentation algorithms and provide a small testing corpus of pre-segmented documents. Dialectic then benchmarks its library of segmentation algorithms and recommends the one with the highest WindowDiff score.

Predicting Low-quality Text

Dialectic reduces the task of supporting new domains to adding new domain-specific feature extraction methods¹ and mapping features predictive of low quality text to explanation text (next subsection). Dialectic implements a library of state-of-the-art features to help bootstrap new deployments. From a survey of existing literature, we identify and implement a diverse library of 47 features summarized in Table 1 (Krause 2015a; Liu et al. 2007; Tan et al. 2016; Siersdorfer et al. 2010; Ghose and Ipeirotis 2011). The primary features that we do not include are those that rely on application metadata such as the author’s history, which are predictive of quality but not related to the writing content.

Dialectic uses the feature library to build two models to predict the quality of individual segments generated by the *Segmenter*, and for the full document. Following Ghose et al. (Ghose and Ipeirotis 2011) we use a random forest classifier, and we exploit its structure for our perturbation-based feedback explanation technique. Given the labeled corpus of documents, Dialectic trains both the segment and document-level classifiers. For each model, we run recursive feature elimination to select the optimal subset of features for the segment and document models (Guyon et al. 2002).

Our model performs competitively with prior work, which predicts the quality of Amazon DVD, AV player and Camera reviews with 83% accuracy when not using metadata features (Ghose and Ipeirotis 2011). Dialectic’s default model on the same setup predicts at 85% accuracy—the slight improvement is due to the additional features in the of topic and similarity categories from other literature (Table 1). We validated the model on reddit comments from the AskScience subreddit² and predicted comment helpfulness on an evenly balanced sample with 80% accuracy³.

One challenge is training the segment-level classifier because most corpi are labeled at the document level. In the **Dialectic Setup** section of the evaluation section, we show an effective default is to train the segment-classifier using document labels as a proxy (i.e. a segment from an 80% helpful document is assigned an 80% helpful training label).

Feedback Generator

Once the models identify low-quality text (a segment or the whole document), they must generate specific feedback to help users improve their writing. Dialectic first uses a novel perturbation-based analysis inspired by (Krause, Perer, and Ng 2016) to estimate the amount that the document’s feature vector needs to be perturbed in order to be classified as high quality. This is used to assign each feature an “impact-score” that represents the amount that it can improve the quality score when perturbed simultaneously with other features. These scores are then used to select the explanation functions that generate document/segment-specific feedback text.

¹Functions that transform text to a fixed-size numeric vector.

²<https://www.reddit.com/r/askscience/>

³We define > 1 net up-votes as helpful and ≤ 1 as unhelpful.

Category	#	Description
Informativeness	8	mined jargon word and named entity stats (Minqing Hu 2004), length measures (word, sentence, etc. count)
Topic	5	LDA topic distribution and top topics (Blei, Ng, and Jordan 2003), entropy across topic distribution
Subjectivity	15	opinion sentence distribution stats (Minqing Hu 2004), valence, polarity, and subjectivity scores and distribution across sentences (Ghose and Ipeirotis 2011; Gilbert 2014; Loria 2014), % upper case characters, first person usage, adjectives
Readability	15	spelling errors (Kelly 2011), ARI, Gunning index, Coleman-Liau index, Flesch Reading tests, SMOG, punctuation, parts of speech distribution, lexical diversity measures
Similarity	4	various TF-IDF and top parts of speech comparisons with sample of low and high utility documents

Table 1: Summary of default feature library (full list will be available in technical report)

Perturbation-based Feature Impact Our perturbation-based analysis extends prior work by Krause et. al (Krause, Perer, and Ng 2016), which perturbs each feature in isolation to estimate the prediction’s sensitivity to each feature. However, this ignores complex patterns in writing quality that emerge from multi-feature interactions. For this reason, we use non-linear random forest models that are capable of modeling these complex feature interactions (Weimer, Gurevych, and Mühlhäuser 2007; Ghose and Ipeirotis 2011). Dialectic estimates the amount that perturbing *sets* of features have on model predictions for a document’s feature vector, which can be done efficiently by exploiting the structure of random forest models. Below, we formalize the model-independent problem of feedback generation, and describe a heuristic solution based on random forests. The **Evaluation** section compare this approach to existing techniques that examine features in isolation (Krause 2015b).

Problem Setup: Let \mathcal{F} be the set of n model features, and f_i denote the i^{th} feature. Let $d \in \mathbb{R}^n$ be a data point (text document or segment) represented as a feature vector, where d_i corresponds to the value of f_i . For instance, \mathcal{F} may be the text features described above, and a data point corresponds to the extracted text feature vector. A model $M : \mathbb{R}^n \rightarrow \mathbb{N}$ classifies data points as $M(d) \in \mathbb{N}$, and a utility function $U : \mathbb{N} \rightarrow \mathbb{R}$ maps a label to a utility score. A perturbation $p \in \mathbb{R}^n$ is a vector that modifies a data point, for a set of features. $p_i \in \mathbb{R} - \{0\}$ if f_i is a perturbed feature in the set, otherwise $p_i = 0$.

Our goal is to identify feature subsets of the test data point d that, if perturbed, will most improve d ’s utility⁴. To do so, we first define the *impact* $I(d, p)$ for an individual perturbation p as the amount that it improves the utility function discounted by the amount of the perturbation $\Delta(p)$ and the model’s prediction confidence $C(d + p) \in [0, 1]$.

S_i^d computes the overall score for feature f_i based on the impact of all perturbations involving f_i :

$$I(d, p) = \frac{U(M(d + p)) - U(M(d))}{\Delta(p)} \times C(d + p)$$

$$\Delta(p) = \sum_{i \in [0, n]} dist(p_i)$$

$$S_i^d = \sum_{p \in \mathbb{R}^n, p_i \neq 0} I(d, p)$$

⁴No feedback needed if data point already has high utility.

By default Dialectic uses the hamming distance as the $dist()$ in the Δ function, and computes C as the percentage of trees that vote for the majority label.

Problem 1 Given datapoint d , compute S_i^d for all features \mathcal{F}

Heuristic Solution: The space of solutions for Problem 1 is exponential in the number of features used by the model, because the cardinality of the power set $|\mathcal{P}(\mathcal{F})| = 2^{|\mathcal{F}|}$, meaning that for n features there are 2^n possible sets of perturbations to naively explore. We instead present a heuristic solution whose complexity is linear in the number of paths in the random forest model. The main idea is to scan each tree in the random forest and compute perturbations and scores local to the tree.

Let the $D = \{d_1, \dots, d_m\}$ be the training dataset and $Y = \{y_1, \dots, y_m\}$ be their labels. The random forest model $M = \{T_1, \dots, T_t\}$ is composed of a set of trees. A tree T_i is composed of a set of k decision paths q_i^1, \dots, q_i^k ; each path q_i^j matches a subset of the training dataset $D_i^j \subseteq D$ and its vote v_i^j is the majority label in D_i^j . Thus, the output of $T_i(d)$ is the vote of the path that matches d , and the output of the random forest $M(d)$ is the majority vote of its trees.

Let $minp(d, q_i^m)$ return the minimum perturbation p (based on its L2 norm) such that d matches path q_i^m :

$$minp(d, q_i^m) = \arg \min_{p \in \mathbb{R}^n} |p|_2 \text{ s.t. } q_i^m \text{ matches } d + p$$

Rather than examining all possible perturbations, our heuristic to compute S_i^d restricts the set of perturbations with respect to the decision paths in the trees that increase d ’s utility. The impact function $I()$ is identical, however it takes a path q_i^j as input and internally computes the minimum perturbation $minp(d, q_i^j)$. Finally, we compute the confidence $C(d)$ as the fraction of samples in D_i^j whose labels y_k match the path’s prediction v_i^j .

$$S_i^d = \sum_{T_i \in M} \sum_{q_i^j \in T_i} I(d, q_i^j) \text{ if } U(v_i^j) > U(M(d))$$

$$I(d, q_i^j) = \frac{U(v_i^j) - U(M(d))}{\Delta(minp(d, q_i^j))} \times C(d + minp(d, q_i^j))$$

$$C(d) = \frac{|\{d_k \in D_i^j | y_k = v_i^j\}|}{|D_i^j|}$$

Our implementation indexes all paths in the random forest by their utility. Given d and predict utility $U(M(d))$, we retrieve and scan the paths with higher utility. For each scanned path q , we compute the change in the utility function, discount its value by the minimum perturbation p as well as the path’s confidence. Finally, we add this value to the score of all features perturbed in p . The final scores are used to select from the library of explanation functions. Performing the indexing process during the offline phase allows us compute these scores on the order of $\frac{1}{10}$ seconds.

Explanation Functions We assume that the developer has implemented *explanation functions* $e : \mathcal{F} \times \mathbb{R}^n \rightarrow \text{text}$. An explanation function has two parts, the *text-generation* and *mapped features*. The *text-generation* method takes as input the feature vector, the document text, its segments, and the segment id if appropriate, and returns the feedback text. *Mapped features* ($\subseteq \mathcal{F}$) is a list of features for which a high average perturbation impact score indicates that the explanation should be executed. For example, features relating to readability (i.e ARI, Flesch Reading Tests, misspellings) could be mapped to an explanation function that asks writers to revise their text to be more clear. Developers can then use prior literature to manually map features to explanations (we demonstrate this process in the evaluation section). In future work, we hope to learn mappings, provided a training set of low-quality documents labeled with relevant explanation functions. In practice, developers extend an *Explainer* class, implement a `call()` method that performs the *text-generation*, and implement a `features()` method to return the *mapped features*. The following abbreviated snippet sketches the *Not Enough Detail* function in our evaluation, which recommends product features to include in the review based on the text’s topic distribution and the number of product features detected.

```
class NotEnoughDetail(Explainer):
    def features(self): return ['topics', 'featureCnt', 'length']
    def __call__(self, feats, text, segs, seg_id):
        if feats['featureCnt'] < 10:
            return suggest_new_prod_feats(feats['topics'], text)
```

Selecting Explanation functions We first adjust feature impact scores to reduce bias; features closer to the root will happen to occur in more feature sets and have artificially higher scores. We adjust each feature’s impact score S_i^d by computing a sample mean u_i and standard deviation σ_i for that score from a sample of low-utility documents. We then compute: $Snorm_i^d = \frac{S_i^d - u_i}{\sigma_i}$. We now *score explanation functions* by the average impact scores of their mapped features. This can be represented as a series of fast matrix operations: Let $\vec{s} \in \mathbb{R}^n$ where $\vec{s}_i = Snorm_i^d$ and construct matrix $A \in \mathbb{R}^{m \times n}$ for m explanation functions ($A_{ji} = 1$ if explanation j is mapped to feature i , otherwise 0). Compute $\vec{e} \in \mathbb{R}^m = A\vec{s}$. $\frac{\vec{e}_j}{\sum_{i=1}^n A_{ji}}$ is the average impact score of all features mapped to the j th explanation function. We then select and return the concatenated text outputs for explanation functions with score $> V$ (a threshold). We manually set V in our evaluation, though it may be estimated by sampling

the distribution of low-utility documents explanation scores and using a percentile (i.e $V = 80^{th}$ percentile of scores).

Feedback Interface and Usage

Our default interface can be seen in (Figure 1). Research shows the efficacy of highlighting to guide users (Antwarg, Lavie, and Rokach 2012) and the importance of providing feedback immediately *after* writing (Anderson 2004); we therefore highlight low-quality segments, and only show feedback after the user finishes writing and pushes *Get Feedback*.

Dialectic is designed to require minimal front-end changes. Dialectic provides a javascript library that automatically augments text input elements with feedback support (Figure 1). After running the pipeline on their corpus, users simply include the Dialectic javascript library annotate textbox and form submission HTML elements with special attributes prefixed by `dialectic-`. The library also provides developers with a Javascript API to customize the design and interactions of the feedback interfaces. Dialectic is available as a pip package *anonymized for submission*

Evaluation

We evaluate Dialectic on an existing corpus of Amazon product reviews (McAuley, Pandey, and Leskovec 2015) through a crowdsourced Mechanical Turk study. We compare against a state-of-the-art feedback system (Krause 2015b) along two dimensions—*granularity* and *explanation selection* (see **Experiment Design**).

Dialectic Setup

We first describe how the Dialectic models and explanation functions were configured to run these experiments. The main challenge is that reviews only contain document-level training labels, and we must also train a segment-level classifier.

Model Training: We trained our models using our default library of 47 features on the Amazon review corpus, and used a cut-off of $\geq 60\%$ “helpful” votes as high quality reviews (positive labels) and the rest as low quality (negative labels) based on (Archak, Ghose, and Ipeirotis 2011). The document-level classifier (acc=85%, prec=81.2%, recall=87.9%) was competitive with existing literature (Ghose and Ipeirotis 2011; Archak, Ghose, and Ipeirotis 2011) on a balanced sample of 500 reviews.

For segment-level classifiers, we hypothesized that document quality is sufficiently correlated with segment quality that a document’s label can be used as the labels for segments for training purposes. To validate this, we first trained a segment classifier using this assumption. We then ran a crowdsourced study to label 500 balanced segments (250 each from helpful/unhelpful reviews)⁵. Despite shorter input text, the classifier performed reasonably well at predicting the manual segment labels (acc=73.6%, prec=76.3%,

⁵For space constraints, we simply report the results of this study.

Features	#	Unhelpfulness Reasons	Explanation Function
Informativeness	8	Lack of Information, Not Enough Detail, Too Short	Infer input topics and suggest relevant, unmentioned product features (Minqing Hu 2004) (Uses cached list of mined product features and associated topics from training),
Topic	5	Irrelevant Comments	Infer topics of input text, suggest several topics that have a low porportion within this distribution, and correlate highly with helpful reviews
Subjectivity	15	Overly Emotional/Biased	Identify words that most contribute to the input text’s sentiment score (Gilbert 2014), and recommend revising to be more balanced in their writing.
Readability	15	Poor Writing Style	Ask the author to revise the writing of the segment or overall document to be clearer

Table 2: Summary of the 4 product review explanation functions. Reasons are from (Connors, Mudambi, and Schuff 2011)

recall=69.7%). These results suggest the efficacy of our simple segment-level classifier, though more studies are needed to fully evaluate this hypothesis across other text domains.

Explanation Functions: We setup Dialectic to generate explanations for a broad range of reasons for why a review may be unhelpful. Prior work found that 75% of reasons for unhelpful reviews were covered by (in priority order) overly emotional/biased opinions, lack of information/not enough detail, irrelevant comments, and poor writing style (Connors, Mudambi, and Schuff 2011). These reasons naturally map to 4 of the 5 Dialectic feature categories: *Subjectivity*, *Informativeness*, *Topic*, and *Readability* respectively (Table 1). We created one explanation function for each category (Table 2) and mapped the function manually to model features belonging in that category. For instance, a high perturbation-based ranking for a topic-related feature would be mapped to the “off-topic” explanation function. In future work, we hope to learn feature to explanation function mappings from developers provided examples.

This process illustrates how Dialectic can be extended to new domains—in many cases, there is substantial literature that 1) identifies reasons for low document quality, 2) constructs features based on these reasons, and 3) suggests ways to structure and provide feedback to the writer. One of our contributions is to provide an end-to-end system so that the above types of literature can be easily added in the form of features and explanation functions. Dialectic then uses the features to train models to detect low quality text, and the perturbation-based technique to select and generate the explanations that are suitable for each low quality document, taking into account multi-feature interactions. Our subsequent field experiment demonstrates the improvements in user-generated text that feedback from Dialectic achieves.

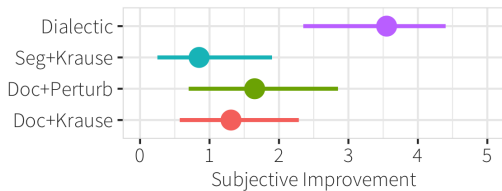


Figure 5: Subjective agreement to: “The post-feedback revisions improved on the pre-feedback review.”

Experiment Design

We evaluated Dialectic through a crowdsourced study on Mechanical Turk. We compared four feedback systems that varied along two dimensions—*granularity* varies the feedback to be at the document level (*Doc*), or at the document *and* segment level (*Seg*), while *explanation selection* compares the technique from (Krause 2015b) (*Krause*) with Dialectic’s perturbation-based explanation selection (*Perturb*). This results in a 2x2 between-subjects design. *Dialectic* denotes the segment-level perturbation-based system.

Krause is based on (Krause 2015b), which was shown to outperform showing writers static explanations of important components of a helpful review (similar to a rubric), in the context of university students writing code-reviews for their peers. *Krause* models the primary qualities of a helpful review (specificity, subjectivity, etc.) using a set of explainable features (i.e document length, emotion). It computes the mean and standard deviation of these features across a corpus of high quality documents. If a user-generated document’s feature value is more than 1.5 SD from the mean, its explanation function is executed. Features are mapped to explanation functions that returns static text about how to improve that feature. Our *Krause* condition implemented the features from Krause as well as domain-specific features that were assigned a high feature weight by Dialectic’s random forest model (# of product features/jargon and Coleman-Liau index to predict specificity and readability respectively). We supplemented the features of *Krause* because the purpose of these experiments was not to show that our specific cocktail of features out-perform prior work, but to show the efficacy of generating segment-specific feedback and the value of our perturbation-based explanation selection.

To summarize, each participant was randomly assigned to one of four conditions: *Doc+Krause*, *Seg+Krause*, *Doc+Perturb* and *Dialectic (Seg+Perturb)*.

Participants: We recruited 85 workers on Amazon’s Mechanical Turk (61.2% male, 38.8% female, ages 20-65 $\mu_{age}=32$, $\sigma_{age}=8.5$). Participants were randomly assigned to one condition group; all conditions had 21 subjects except the *Dialectic* condition which had 22. No participant had used Dialectic before. 71.3% had written a prior product review; all had read a product review in the past. All participants were US Residents with > 90% HIT accept rates. The average task completion time was 14 minutes, and payment was \$2.5 (\sim \$10/hr).

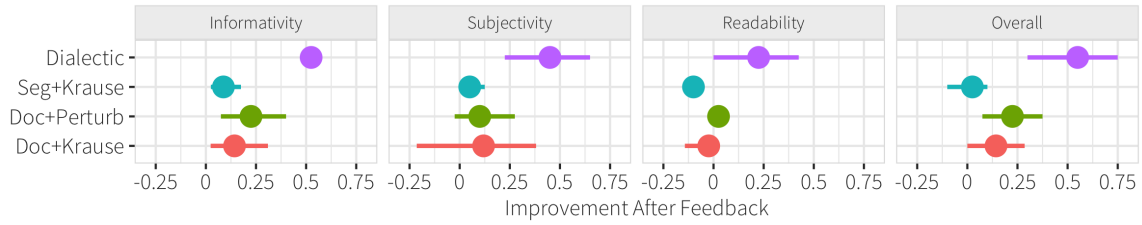


Figure 4: Improvement in informativity, subjectivity, readability, and overall quality Likert scores across all four conditions

Procedure: Participants were asked to write and then revise a review of their most recently owned laptop computer. We used a qualification task to ensure participants had ever owned a laptop. We explained the feedback interface and did not offer a rubric, then asked participants to write their review “as if they are trying to help someone else decide to buy that laptop or not... as they would on a review website like the Amazon store”. The I’m Done Writing button displayed our document-level feedback under the text field; for users in the segmentation condition, low quality segments were highlighted red and the related feedback displayed when users hovered over the segment. We then gave participants the opportunity to revise their review; to avoid bias, we noted that they were **not** obligated to. At this point, users could click the Recompute Text Feedback button (median 1 click/participant), or press Submit to submit and finish the task. We used a post-study survey to collect demographic information as well as their subjective experience.

The interface was the same for all conditions—only the feedback content changed. The final submission was considered the *post-feedback* review, and the initial submission upon pressing the I’m Done Writing was the *pre-feedback* review. The experiment was IRB approved.

Results

Review Evaluation: 81 of the 85 participants completed the review writing task. Three independent evaluators coded the pre and post-feedback reviews using a rubric based on prior work on review quality (Connors, Mudambi, and Schuff 2011; Mudambi and Schuff 2010; Liu et al. 2007). The rubric rated reviews on a 1-7 Likert scale using three specific aspects—informativity, subjectivity, readability—as well as a holistic overall score. The change in these measures between pre and post-feedback suggests the utility of the feedback.

Finally, we asked reviewers to subjectively rate their agreement from 1-7 to the statement “The post-feedback revisions improved on the pre-feedback review.”, or 0 if the review did not change. Each measure is the average of the ratings from two coders—if they differed by > 3 , a third expert coder was used as the tie breaker and decided the final value. The third coder was trained by being shown the Amazon review corpus, examples across the quality spectrum, and the other two coders. The coders labeled reviews

in random order and did not have access to any other information about the text.

Rubric Description: The rubric asks coders to score reviews on helpfulness to laptop shoppers. It defines the three main measures, and provides examples from the Amazon corpus that contribute positively and negatively to each criterion. *Informativity* is the extent that the review provides detailed information about the product, where 7 means that the review elaborates on all or almost all of the specifications of a product while 1 means that it states an opinion but fails to provide factual details (e.g., laptop specifications). *Subjectivity* is the extent that the review is fair and balanced but with enough helpful opinions for the buyer to make an informed decision: 1 means the review is an angry rant or lacks any opinions while 7 means it is a fair and balanced opinion. *Readability* is the extent that the review facilitates or obfuscates the writer’s meaning. For instance, a review that consists of many ambiguous phrases like “I have never done anything crazy with it and it still works.” is assigned 1 as it might require multiple readings to understand. *Overall Quality* is the holistic helpfulness of the review for prospective buyers.

Analysis

Figure 4 plots the mean change and 95% bootstrap confidence interval for the four rubric scores. Figure 5 shows a similar chart for the coder’s subjective opinion of the improvement. These plots show the effect size across all measures, and that the largest improvements were due to the combination of segmentation and perturbation-based explanation. We now focus our analysis on the change in overall quality metric (right-most facet Figure 4). Analogous statistical tests on the data in Figure 5 produced the same conclusions.

A one-way ANOVA first showed that the four conditions had a significant influence on the overall change in quality ($F(3,77)=7.11, p<1e^{-4}$). Using Tukey’s HSD post-hoc test to compare the individual conditions, we found that the pairwise comparisons between the *Dialectic* condition ($\mu=0.55, \sigma=0.51$) and the other three conditions *Doc+Perturb* ($\mu=0.23, \sigma=0.34$), *Seg+Krause* ($\mu=0.025, \sigma=0.26$), *Doc+Seg* ($\mu=0.14, \sigma=0.51$) were significant ($p<0.05$). However, all pairwise comparisons between the latter three conditions did not show significance.

We then performed a two-way ANOVA using overall quality increase as the dependent variable, and per-

turbation and segmentation as the independent variables. The ANOVA found a significant effect for perturbation ($F(1,80)=9.66$, $p=0.0026<.005$). We can see this in Figure 4: Dialectic outperformed *Seg+Generic*, and *Doc+Perturb* outperformed *Doc+Krause*. On the other hand, segmentation alone did not have a significant effect ($F(1,80)=2.21$, $p=0.14$). Finally, interaction effects between segmentation & perturbation were significant ($F(1,80)=5.75$, $p=0.019<.05$) which can be seen from the Tukey HSD test where adding segmentation to *Doc+Perturb* greatly improved overall review quality and adding segmentation to *Doc+Krause* had a negative effect.

In summary, combining segmentation and perturbation-based explanation outperformed all other conditions by a statistically significant margin. Controlling for the other variable, perturbation-analysis was shown to cause a statistically significant improvement in improvement, while segmentation did not. However, the effect sizes of both variables in isolation were relatively small compared to the effect of combining them; Dialectic, which combines segmentation and perturbation-analysis, improved the overall measure (right-most facet) by nearly $3.9\times$ over the baseline (0.55 vs. 0.14 increase), and a $2.4\times$ improvement over the next-best *Doc+Perturb* condition. Significant interactions effects suggest this to be due to a co-dependency between segmentation and perturbation methods; granular feedback is not useful when a model does not provide complex insights into each segment, and the perturbation-analysis insights are less useful when constrained to the document level. All feedback was generated within one second.

Participant Feedback: The post-trial questionnaire asked participants to indicate their level of agreement with a few statements regarding their experience on a 7-point Likert scale (1 - strongly disagree, 7 - strongly agree). Participants agreed with the statement “*The interface was easy to use*” for both the baseline *Doc+Krause* ($\mu=5.81$, $\sigma=1.33$) and *Dialectic* ($\mu=5.82$, $\sigma=1.18$) conditions. This suggests that the interface design facilitated the participants’ writing. However, in response to “*Using the interface improved my review*,” there was a statistically significant ($t(39)=2.998$, $p=0.027<.05$) difference between *Doc+Krause* ($\mu=3.81$, $\sigma=1.86$) and *Dialectic* ($\mu=5.00$, $\sigma=1.41$), suggesting that Dialectic’s segment-oriented feedback contributed to improved writing. Regarding the targeted and segment level feedback of Dialectic, users praised it for making them reconsider specific aspects of their writing; e.g. “*the feedback I got was over parts where I wrote [sic] alot... it seemed like unnecessary fluff after giving it a second read when it was highlighted in the feedback, and I ended up taking some parts of it out.*” Conversely, when asked to describe the suggestions from *Doc+Krause*, one user wrote “*It was general feedback nothing specific,*” and another said they were “*not sure of what needed to change.*” Overall, these comments suggest that there is value in providing segment-oriented feedback and that explanation functions helped produce more actionable feedback by taking into account multiple features as well as the input text.

The most widespread criticism regarding Dialectic was after users had edited their reviews after the first round of feedback and then pressed *Recompute Text Feedback* a second time. If the edits did not address the predicted issues, Dialectic would generate the same feedback, which was not productive. For example, one user wrote: “*Initially it seemed interesting but even after editing... nothing changed when I resubmitted.*” This was understandable given that we only used four simple explanation functions. Extensions that take into account prior feedback may help ameliorate these complaints.

Finally, we found that a few ($N=3$) users were reluctant to trust computer-generated feedback; one stated that they didn’t make changes because “*It seemed like the feedback was just an automated response.*” Creating more customized and human-like feedback is a promising area of future work.

Conclusion and Future Work

This paper presented the design, implementation and evaluation of an extensible feedback system for user text inputs. We demonstrate that by combining segmentation, classification, and explanation techniques, it is possible to create automated interfaces that improve the quality of user generated social content. Moreover, we demonstrate that the creation of such interfaces can be streamlined to require minimal effort on the part of developers. Through a crowd study, we find that Dialectic is able to improve the total writing quality score of laptop reviews by 14.4%, over $3\times$ a state-of-the-art system.

Though Dialectic demonstrates the feasibility of such automated interfaces, it also reveals several areas of improvement. Due to a small (4) number of explanation functions, study participants found that repeatedly using the system began to provide redundant feedback; simplifying the development of more explanation functions may help the system produce more nuanced feedback. We also used document-quality labels to train the segment classifier. We showed this to be sufficient by testing on crowd-sourced labels; however more sophisticated techniques to classify segments could improve feedback. Finally, we hope to explore applications of Dialectic to different social media domains and different user contexts.

In the long term, we envision Dialectic as an example of automatically applying input-side optimizations (i.e writing feedback) based on downstream application needs (i.e quality reviews). In many cases, input-side optimizations can improve the quality of the application or drastically reduce costs of downstream processes. For instance, entity resolution (Getoor and Machanavajjhala 2012) is often performed by social media services at a cost quadratic with respect to the dataset size. If data is de-duplicated during input, we can eliminate such expensive procedures. We hope to explore other methods of augmenting the input-side upstream in future work.

References

Agichtein, E.; Castillo, C.; Donato, D.; Gionis, A.; and Mishne, G. 2008. Finding high-quality content in social media. In *WSDM*.

- Amazon. 2016. Amazon: Community guidelines. <https://www.amazon.com/gp/help/customer/display.html?nodeId=201929730>.
- Anderson, T. 2004. Teaching in an online learning context. volume 273.
- Antwarg, L.; Lavie, T.; and Rokach, L. 2012. Highlighting items as means of adaptive assistance. In *Behavior and Information Technology*.
- Archak, N.; Ghose, A.; and Ipeirotis, P. G. 2011. Deriving the pricing power of product features by mining consumer reviews. In *Management Science*. INFORMS.
- Attali, Y., and Burstein, J. 2004. Automated essay scoring with e-rater® v. 2.0. In *ETS Research Report Series*. Wiley Online Library.
- Bakshy, E.; Karrer, B.; and Adamic, L. A. 2009. Social influence and the diffusion of user-created content. In *EC*.
- Biran, O., and McKeown, K. 2014. Justification narratives for individual classifications. In *AutoML*.
- Blei, D. M.; Ng, A. Y.; and Jordan, M. I. 2003. Latent dirichlet allocation. In *JMLR*.
- Boomerang. 2016. Responsible: Personal ai assistant for writing better emails. <http://www.boomeranggmail.com/responsible/>.
- Bosu, A.; Corley, C. S.; Heaton, D.; Chatterji, D.; Carver, J. C.; and Kraft, N. A. 2013. Building reputation in stackoverflow: an empirical investigation. In *MSR*.
- Connors, L.; Mudambi, S. M.; and Schuff, D. 2011. Is it the review or the reviewer? a multi-method approach to determine the antecedents of online review helpfulness. In *System Sciences (HICSS), 2011 44th Hawaii International Conference on*, 1–10. IEEE.
- Deterding, S.; Dixon, D.; Khaled, R.; and Nacke, L. 2011. From game design elements to gamefulness: defining gamification. In *MindTrek*.
- Farra, N.; Somasundaran, S.; and Burstein, J. 2015. Scoring persuasive essays using opinions and their targets. In *NAACL*.
- FoxType. 2016. Write smarter emails. foxtype.com/.
- Getoor, L., and Machanavajjhala, A. 2012. Entity resolution: theory, practice & open challenges. *Vldb*.
- Ghose, A., and Ipeirotis, P. 2009. The economining project at nyu: Studying the economic value of user-generated content on the internet. In *Journal of Revenue & Pricing Management*. Nature Publishing Group.
- Ghose, A., and Ipeirotis, P. G. 2011. Estimating the helpfulness and economic impact of product reviews: Mining text and reviewer characteristics. In *TKDE*. IEEE.
- Ghose, A.; Ipeirotis, P. G.; and Sundararajan, A. 2007. Opinion mining using econometrics: A case study on reputation systems. In *ACL*.
- Ghosh, A. 2012. Social computing and user-generated content: a game-theoretic approach. In *ACM SIGecom Exchanges*. ACM.
- Gilbert, C. H. E. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text.
- Google. 2016. Check spelling and grammar in google docs. support.google.com/docs/answer/57859.
- Guy, I. 2015. Social recommender systems. In *Recommender Systems Handbook*. Springer.
- Guyon, I.; Weston, J.; Barnhill, S.; and Vapnik, V. 2002. Gene selection for cancer classification using support vector machines. In *Machine learning*. Springer.
- Hearst, M. A. 1997. Texttiling: Segmenting text into multi-paragraph subtopic passages. In *Computational Linguistics*. MIT Press.
- Hyönä, J.; Lorch Jr, R. F.; and Kaakinen, J. K. 2002. Individual differences in reading to summarize expository text: Evidence from eye fixation patterns. volume 94, 44. American Psychological Association.
- Ipeirotis, P. 2016. Fix reviews' grammar, improve sales. behind-the-enemy-lines.com/2011/04/want-to-improve-sales-fix-grammar-and.html.
- Ji-Wei Wu, J. C. T. 2011. An efficient linear text segmentation algorithm using hierarchical agglomerative clustering. In *CIS*.
- Kelly, R. 2011. r.fk/pyenchant.
- Kim, S.-M.; Pantel, P.; Chklovski, T.; and Pennacchiotti, M. 2006. Automatically assessing review helpfulness. In *ACL*.
- Kim, J. 2012. The institutionalization of youtube: From user-generated content to professionally generated content. In *Media, Culture & Society*. Sage Publications.
- Krause, J.; Perer, A.; and Ng, K. 2016. Interacting with predictions: Visual inspection of black-box machine learning models. In *HCI*.
- Krause, M. 2015a. Bull-o-meter: Predicting the quality of natural language responses. In *HCOMP*.
- Krause, M. 2015b. A method to automatically choose suggestions to improve perceived quality of peer reviews based on linguistic features.
- Kulik, J. A., and Kulik, C.-L. C. 1988. Timing of feedback and verbal learning.
- Kulkarni, C. E.; Bernstein, M. S.; and Klemmer, S. R. 2015. Peer-studio: Rapid peer feedback emphasizes revision and improves performance. In *ACM*.
- Lei, T.; Barzilay, R.; and Jaakkola, T. S. 2016. Rationalizing neural predictions. In *EMNLP*.
- Li, B.; Ghose, A.; and Ipeirotis, P. G. 2011. Towards a theory model for product search. In *WWW*.
- Liu, J.; Cao, Y.; Lin, C.-Y.; Huang, Y.; and Zhou, M. 2007. Low-quality product review detection in opinion summarization. In *EMNLP-CoNLL*.
- Liu, Y.; Huang, X.; An, A.; and Yu, X. 2008. Modeling and predicting the helpfulness of online reviews. In *IEEE Computer Society*.
- Loria, S. 2014. Textblob: Simplified text processing.
- Madnani, N., and Cahill, A. 2014. An explicit feedback system for preposition errors based on wikipedia revisions. In *NAACL*.
- McAuley, J.; Pandey, R.; and Leskovec, J. 2015. Inferring networks of substitutable and complementary products. In *KDD*.
- Microsoft. 2016. Check spelling and grammar in office 2010 and later. support.office.com.
- Minqing Hu, B. L. 2004. Mining opinion features in customer reviews. In *AAAI*.
- Misra, H.; Yvon, F.; Cappé, O.; and Jose, J. 2011. Text segmentation: A topic modeling perspective. In *Information Processing & Management*. Elsevier.
- Muchnik, L.; Aral, S.; and Taylor, S. J. 2013. Social influence bias: A randomized experiment. In *Science*. American Association for the Advancement of Science.
- Mudambi, S. M., and Schuff, D. 2010. What makes a helpful review? a study of customer reviews on amazon. com. In *MIS quarterly*.

- Nelson, M., and Schunn, C. 2007. The nature of feedback: Investigating how different types of feedback affect writing performance. In *Learning Research and Development Center*.
- Nov, O. 2007. What motivates wikipedians? In *Communications of the ACM*. ACM.
- Pevzner, L., and Hearst, M. A. 2002. A critique and improvement of an evaluation metric for text segmentation. In *Computational Linguistics*. MIT Press.
- Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2016. Model-agnostic interpretability of machine learning.
- Riedl, M., and Biemann, C. 2012. Topictiling: a text segmentation algorithm based on lda. In *ACL*.
- Rivers, K., and Koedinger, K. R. 2014. Automating hint generation with solution space path construction. In *ICITS*.
- Roy Choudhury, R.; Yin, H.; and Fox, A. 2016. Scale-driven automatic hint generation for coding style. In *Springer-Verlag*.
- Saito, G. 2016. Unanswered quora. quora.com/What-percentage-of-questions-on-Quora-have-no-answers.
- Siersdorfer, S.; Chelaru, S.; Nejd, W.; and San Pedro, J. 2010. How useful are your comments?: analyzing and predicting youtube comments and comment ratings. In *WWW*.
- Singh, R.; Gulwani, S.; and Solar-Lezama, A. 2012. Automated semantic grading of programs. Technical report, MIT.
- Spirin, N., and Han, J. 2012. Survey on web spam detection: principles and algorithms. In *KDD*. ACM.
- Tan, C.; Niculae, V.; Danescu-Niculescu-Mizil, C.; and Lee, L. 2016. Winning arguments: Interaction dynamics and persuasion strategies in good-faith online discussions. In *WWW*.
- Tang, J.; Hu, X.; and Liu, H. 2013. Social recommendation: a review. In *SNAM*. Springer.
- Valenti, S.; Neri, F.; and Cucchiarelli, R. 2003. An overview of current research on automated essay grading. In *Journal of Information Technology Education*.
- Wang, G.; Gill, K.; Mohanlal, M.; Zheng, H.; and Zhao, B. Y. 2013. Wisdom in the social crowd: an analysis of quora. In *WWW*.
- Weimer, M.; Gurevych, I.; and Mühlhäuser, M. 2007. Automatically assessing the post quality in online discussions on software. In *ACL*.
- Wikipedia. 2016. Editor guidelines. en.wikipedia.org/wiki/Wikipedia:Policies_and_guidelines.
- Yang, L., and Amatriain, X. 2016. Recommending the world's knowledge: Application of recommender systems at quora. In *RecSys*.
- Yelp. 2016a. Applying to be elite. yelp.com/elite.
- Yelp. 2016b. Content guidelines. yelp.com/guidelines.
- Zappos. 2016. What in the hay is a zappos premier reviewer? <http://www.zappos.com/premier-reviewers>.